案例 7-直线反走样 Wu 算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

直线反走样 Wu 算法：直线反走样算法是在 Bresenham 算法的基础上发展的，直线段反走样是从前景色变化到背景色的一个过程，反走样算法将 Bresenham 算法中的“中点误差项”变为了“距离误差项”，“距离误差项”是“中点误差项”的一个具体应用；在绘制反走样直线时，利用不同像素单元格相对于背景色的深浅的不同，通过距离误差项使该像素格附近的像素单元格的颜色形成色差，与背景色平滑过渡；像素单元格的上下两个点的像素的颜色分别为 $C_d = (C_b - C_f) * e_i + C_f$, $C_u = (C_b - C_f) * (1 - e_i) + C_f$,前景色的线性插值表示为 $c = (1 - t) * c0 + t * c_1$,c_0 为直线段的起点颜色，c_1 为直线段的终点颜色。

二、案例描述

本案例以屏幕中心为圆点，绘制一系列相差指定角度的反走样直线，围成一个圆，并为反走样直线设置了两个像素的宽度，以增加直线的平滑程度。

三、实现步骤

1. 在 CTestView 类中添加 DoubleBuffer()双缓冲函数。
2. 添加 CRGB 类和 CP2i 类。
3. 添加 CALine 类，在该类中绘制反走样直线。
4. 在 CTestView 类中添加 DrawObject 函数，绘制直线构成的圆。
5. 在 CTestView 类的构造函数中对变量进行初始化。
6. 在 OnDraw()中对 SetTimer()和 DoubleBuffer()进行调用。

四、主要算法

1.CALine 类

```
public:
    CALine();
    virtual ~CALine();
    void SetBkClr(CRGB bkclr);//设置背景色函数
    void AMoveTo(CDC* pDC, CP2i p0);//移动到指定位置函数
    void AMoveTo(CDC* pDC, int x0, int y0);//重载函数
    void AMoveTo(CDC* pDC, int x0, int y0, CRGB c0);
    void ALineTo(CDC* pDC, CP2i p1);//绘制反走样直线函数，不含终点
    void ALineTo(CDC* pDC, int x1, int y1);
    void ALineTo(CDC* pDC, int x1, int y1,CRGB c1);
    CRGB Interpolation(int m, int m0, int m1, CRGB c0, CRGB c1);
public:
    CP2i P0;//起点
    CP2i P1;//终点
    CRGB BkClr;//屏幕背景色
void CALine::ALineTo(CDC* pDC, CP2i p1)//绘制反走样直线函数
{
    P1 = p1;
    CP2i p,t;
    CRGB c0,c1;
    if (P0.x == P1.x)//绘制垂线
    {
        if (P0.y < P1.y)
        {
            for (p=P0; p.y < P1.y; p.y++)
            {
                p.c = Interpolation(p.y, P0.y, P1.y, P0.c, P1.c);
                pDC->SetPixelV(p.x,p.y,
                    RGB(p.c.red * 255, p.c.green * 255, p.c.blue * 255));
            }
        }
        else
        {
            for (p = P0; p.y > P1.y; p.y--)
            {
                p.c = Interpolation(p.y,P0.y,P1.y,P0.c,P1.c);
                pDC->SetPixelV(p.x,p.y,
                    RGB(p.c.red*255,p.c.green*255,p.c.blue*255));
            }
        }
    }
```

```
else
{
    double k,e = 0;//k 为直线斜率,e 为交点到像素点的距离
    k = double(P1.y - P0.y) / (P1.x - P0.x);
    if (k > 1.0)//绘制 k>1
    {
        if (P0.y < P1.y)
        {
            for (p = P0;p.y < P1.y; p.y++)
            {
                p.c = Interpolation(p.y, P0.y, P1.y, P0.c,P1.c);
                c0 = (BkClr - p.c) * e + p.c;
                c1 = (BkClr - p.c)* (1 - e) + p.c;
                pDC->SetPixelV(p.x, p.y,
                     RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
                pDC->SetPixelV(p.x + 1, p.y,
                     RGB(c1.red * 255, c1.green * 255, c1.blue * 255));
                e = e + 1 / k;
                if (e >= 1.0)
                {
                    p.x++;
                    e--;
                }
            }
        }
        else
        {
            for (p = P0; p.y > P1.y; p.y--)
            {
                p.c = Interpolation(p.y, P0.y, P1.y, P0.c, P1.c);
                c0 = (BkClr - p.c) * e + p.c;
                c1 = (BkClr - p.c) * (1 - e) + p.c;
                pDC->SetPixelV(p.x, p.y,
                     RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
                pDC->SetPixelV(p.x - 1, p.y,
                     RGB(c1.red*255, c1.green * 255, c1.blue * 255));
                e = e + 1 / k;
                if (e >= 1.0)
                {
                    p.x--;
                    e--;
                }
            }
        }
```

```
}
if (0.0 <= k && k <= 1.0)//绘制 0≤k≤1
{
    if (P0.x < P1.x)
    {
        for (p = P0; p.x < P1.x; p.x++)
        {
            p.c = Interpolation(p.x, P0.x, P1.x, P0.c, P1.c);
            c0 = (BkClr - p.c) * e + p.c;
            c1 = (BkClr - p.c) * (1 - e) + p.c;
            pDC->SetPixelV(p.x, p.y,
                 RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
            pDC->SetPixelV(p.x, p.y + 1,
                 RGB(c1.red*255, c1.green * 255, c1.blue * 255));
                e = e + k;
            if(e >= 1.0)
            {
                p.y++;
                e--;
            }
        }
    }
    else
    {
        for (p = P0; p.x > P1.x; p.x--)
        {
            p.c = Interpolation(p.x, P0.x, P1.x, P0.c, P1.c);
            c0 = (BkClr - p.c) * e + p.c;
            c1 = (BkClr - p.c) * (1 - e) + p.c;
            pDC->SetPixelV(p.x, p.y,
                 RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
            pDC->SetPixelV(p.x, p.y - 1,
             RGB(c1.red * 255, c1.green * 255, c1.blue * 255));
            e = e + k;
            if (e >= 1.0)
            {
                p.y--;
                e--;
            }
        }
    }
}
if (k >= -1.0 && k < 0.0)//绘制-1≤k＜0
{
```

```
    if (P0.x < P1.x)
    {
        for (p = P0; p.x < P1.x; p.x++)
        {
            p.c =Interpolation(p.x, P0.x,P1.x, P0.c, P1.c);
            c0 = (BkClr - p.c) * e + p.c;
            c1 = (BkClr - p.c) * (1 - e) + p.c;
            pDC->SetPixelV(p.x, p.y,
                 RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
            pDC->SetPixelV(p.x, p.y - 1,
                 RGB(c1.red * 255, c1.green * 255, c1.blue * 255));
            e = e - k;
            if (e >= 1.0)
            {
                p.y--;
                e--;
            }
        }
    }
    else
    {
        for(p = P0; p.x > P1.x; p.x--)
        {
            p.c =Interpolation(p.x, P0.x, P1.x, P0.c, P1.c);
            c0 = (BkClr - p.c) * e + p.c;
            c1 = (BkClr - p.c) * (1 - e) + p.c;
            pDC->SetPixelV(p.x, p.y,
                 RGB(c0.red *255, c0.green * 255, c0.blue * 255));
            pDC->SetPixelV(p.x, p.y + 1,
                 RGB(c1.red * 255, c1.green * 255,c1.blue * 255));
            e = e-k;
            if (e >= 1.0)
            {
                p.y++;
                e--;
            }
        }
    }
}
if (k < -1.0)//绘制 k<-1
{
    if (P0.y > P1.y)
    {
        for (p = P0; p.y > P1.y; p.y--)
```

```
                {
                    p.c = Interpolation(p.y, P0.y, P1.y, P0.c, P1.c);
                    c0 = (BkClr - p.c) * e + p.c;
                    c1 = (BkClr - p.c) * (1 - e)+p.c;
                    pDC->SetPixelV(p.x, p.y,
                         RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
                    pDC->SetPixelV(p.x + 1, p.y,
                         RGB(c1.red * 255,c1.green * 255,c1.blue * 255));
                    e = e - 1 / k;
                    if(e >= 1.0)
                    {
                        p.x++;
                        e--;
                    }
                }
            }
            else
            {
                for (p = P0; p.y < P1.y; p.y++)
                {
                    p.c = Interpolation(p.y, P0.y, P1.y, P0.c, P1.c);
                    c0 = (BkClr - p.c) * e + p.c;
                    c1 = (BkClr - p.c) * (1 - e) + p.c;
                    pDC->SetPixelV(p.x, p.y,
                         RGB(c0.red * 255, c0.green * 255, c0.blue * 255));
                    pDC->SetPixelV(p.x - 1, p.y,
                         RGB(c1.red * 255, c1.green * 255, c1.blue * 255));
                    e = e-1 / k;
                    if(e >= 1.0)
                    {
                        p.x--;
                        e--;
                    }
                }
            }
        }
    }
    P0=p1;
}
```

2.CTestView 类

```
public:
    CTestDoc* GetDocument();
    void DrawObject(CDC* pDC);//绘制图形
protected:
```

```
    CP2i* p;//顶点数组
    int nCount;//圆等分点
    CALine *aline;//反走样直线指针对象
    double angle;//旋转角
    BOOL bPlay;//动画开关
    CRGB Bc;//定义屏幕背景色
    CRect rect;//定义矩形
void CTestView::DoubleBuffer(CDC* pDC)
{
    CRect rect;//定义客户区矩形
    GetClientRect(&rect);//获得客户区的大小
    pDC->SetMapMode(MM_ANISOTROPIC);//pDC 自定义坐标系
    pDC->SetWindowExt(rect.Width(), rect.Height());//设置窗口范围
    pDC->SetViewportExt(rect.Width(), -rect.Height());
                                    //设置视区范围,x 轴水平向右，y 轴垂直向上
    pDC->SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
                                    //客户区中心为原点
    CDC memDC;//内存 DC
    CBitmap NewBitmap, *pOldBitmap;//内存中承载的临时位图
    memDC.CreateCompatibleDC(pDC);//创建一个与显示 pDC 兼容的内存 memDC
    NewBitmap.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
                                        //创建兼容位图
    pOldBitmap = memDC.SelectObject(&NewBitmap);//将兼容位图选入 memDC
    memDC.FillSolidRect(rect,CRGBTORGB(Bc));//按原来背景填充客户区，否则是黑色
    memDC.SetMapMode(MM_ANISOTROPIC);//memDC 自定义坐标系
    memDC.SetWindowExt(rect.Width(), rect.Height());
    memDC.SetViewportExt(rect.Width(), -rect.Height());
    memDC.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
    rect.OffsetRect(-rect.Width() / 2, -rect.Height() / 2);
    DrawObject(&memDC);//向 memDC 绘制图形
    pDC->BitBlt(rect.left, rect.top, rect.Width(), rect.Height(), &memDC,
                -rect.Width() / 2, -rect.Height() / 2, SRCCOPY);
                                    //将内存 memDC 中的位图拷贝到显示 pDC 中
    memDC.SelectObject(pOldBitmap);//恢复位图
    NewBitmap.DeleteObject();//删除位图
}
void CTestView::DrawObject(CDC* pDC)
{
    double nRadius = sqrt(450 * 450 + 1000 * 1000);//圆的半径
    for (int i = 0; i < nCount; i++)
    {
        p[i].x = ROUND(nRadius * cos(i * 10 * PI / 180 + angle));
        p[i].y = ROUND(-nRadius * sin(i * 10 * PI / 180 + angle));
    }
```

```
        aline->SetBkClr(Bc);
        pDC->FillSolidRect(rect, CRGBTORGB(Bc));
        for (int i = 0; i < nCount; i++)
        {
            aline->AMoveTo(pDC, p[nCount].x, p[nCount].y, CRGB(1.0, 0.0, 0.0));
            aline->ALineTo(pDC, p[i].x, p[i].y, CRGB(0.0, 0.0, 1.0));
        }
    }
```

五、实现效果

直线反走样 Wu 算法效果如图 7-1 和 7-2 所示。

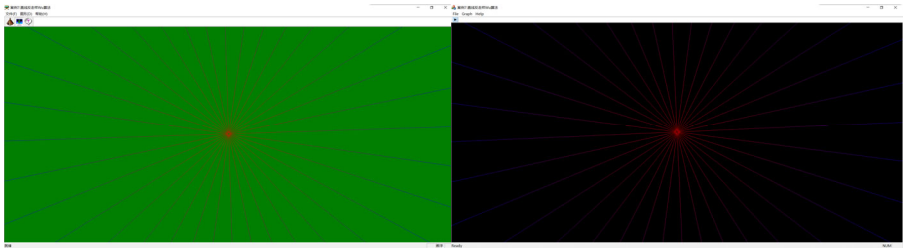

图 7-1 直线反走样 Wu 算法效果图

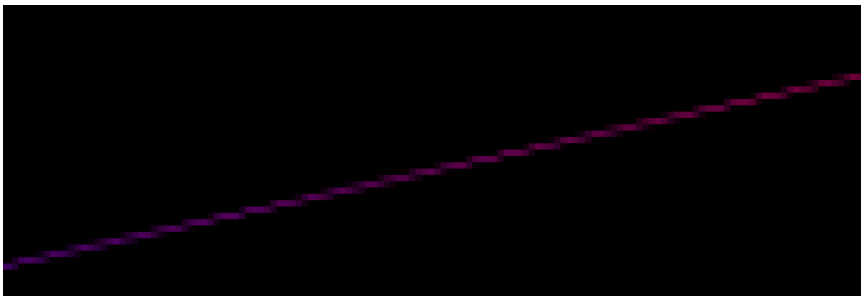

图 7-2 直线反走样 Wu 算法细节图

六、遇到的问题及解决方案

反走样效果与背景色的联系：原始的灰度直线反走样算法只考虑到是黑白两种颜色的过渡，即不考虑背景色与反走样的关系，当背景色改变时，会使得直线失去反走样效果，如果考虑背景色的问题，则彩色直线的反走样是从前景色变化到背景色，形成模糊边界，达到反走样效果，这一点在静态背景和动态背景条件下也都同样适用，核心算法如下：

$$c0 = (BkClr - p.c) * e + p.c;$$

$$c1 = (BkClr - p.c) * (1 - e) + p.c;$$

七、案例心得

通过对直线反走样 Wu 算法的学习，加深了对 Bresenham 算法的理解，了解了中点误差项和距离误差项的联系与区别，并且认识到反走样的程度和直线的像

素有关，如果直线仅为一个像素，则无法反走样，直线像素大于等于 2 个，直线才可以反走样，并且像素越高，反走样效果越好，直线越平滑。